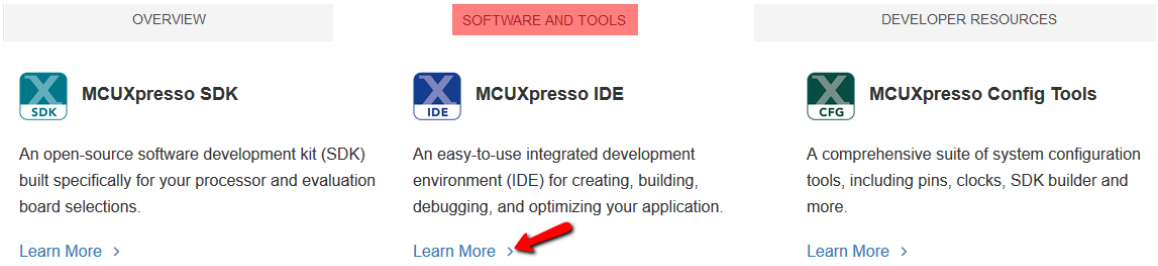
Lab 1 – Create a Thing on AWS

Create an AWS account

1. Open the AWS homepage aws.amazon.com and choose to Create a Free Account
2. Follow the online instructions to create a free account. Part of the sign-up procedure involves:
   2.1. Add payment information, where you will be charged $1.00 USD.
   No additional charges will apply.
   2.2. Receiving a phone call and entering a PIN using your phone's keypad.
3. Wait up to 24 hours for your AWS account to be activated.

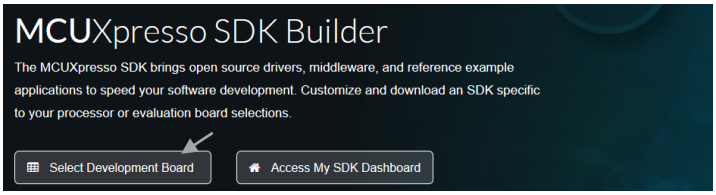
Download MCUXpresso IDE and LPC54018 IoT Module SDK

4. Open a web browser and navigate to the MCUXpresso homepage mcuxpresso.nxp.com
5. Download and install the latest MCUXpresso IDE.
   5.1. Select **Software and Tools;** the click Learn More about MCUXpresso IDE



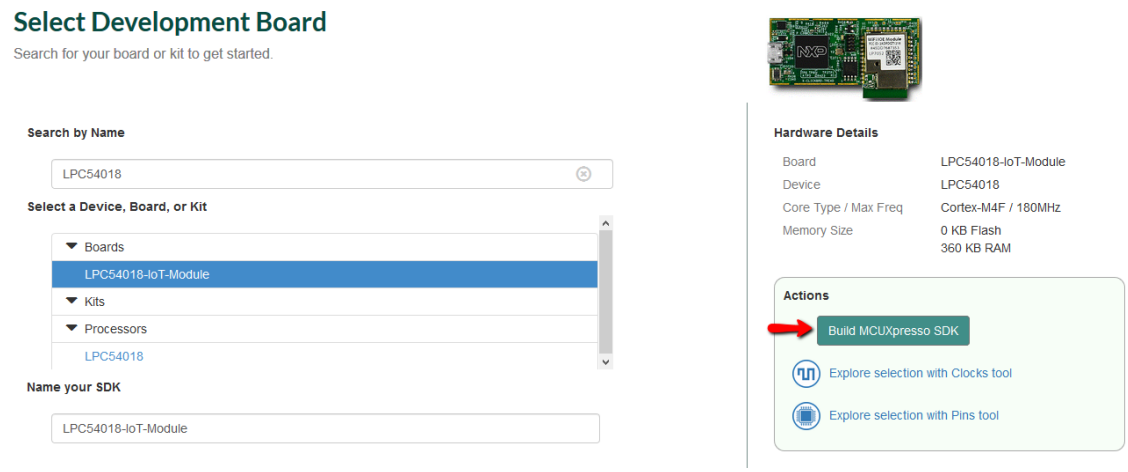
   5.2. Download & Install MCUXpresso IDE

6. Click on Select Development Board to download a customized LPC54018 AWS SDK

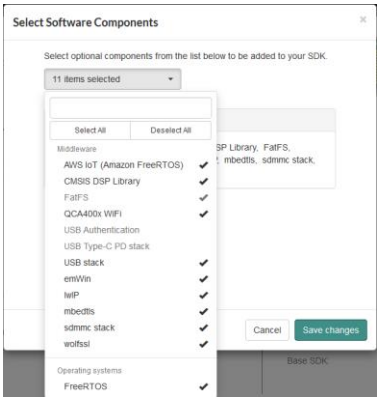


7. After being redirected to nxp.com login page. Enter your account information or register for a new account.

   Back on the Select Development Board page, search the LPC54018. Choose the LPC54018-IoT-Module, then click Build MCUXpresso SDK



8. At the Select Optional Middleware section, click the + Add software component, a window pop-up with a list of components, Select All then click Save Changes

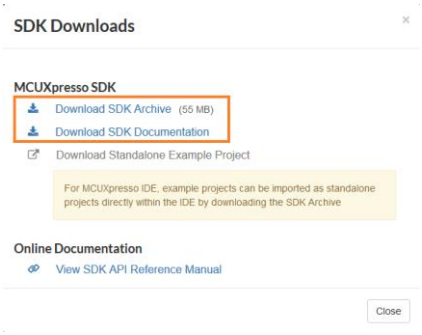


9. Select **Download SDK**

   Note: you may see “Request to Build” instead, so click on that.

10. After the package gets built, there will be a new download in the SDK Dashboard section.

11. Agree to the Software Terms and Conditions, then **Download SDK Archive & SDK Documentation**



Registering a Thing on AWS IoT

12. Log into AWS – Device Console using an IAM account

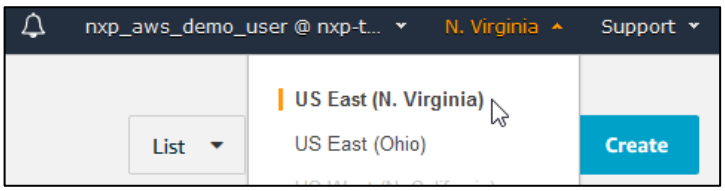
    12.1. Goto: https://aws.amazon.com/

    12.2. Log in with your credentials.

13. Access the **IoT Device Management Console**

    13.1. In the AWS services search, begin typing in "**iot device management**", once the desire results appear in the result window, click the result to access the Management Console

    13.2. Verify that the Region in the upper-right corner is set to "N. Virginia"

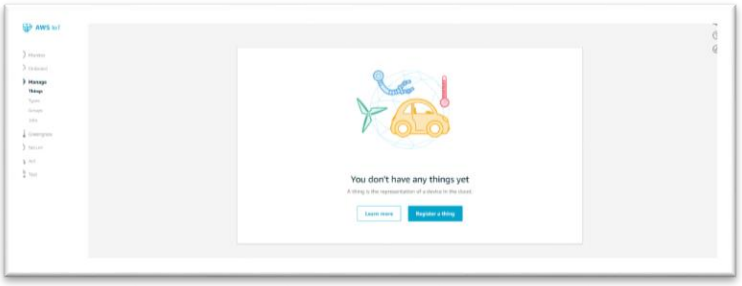


    13.3. Ensure you are on the **Manage -> Things** page

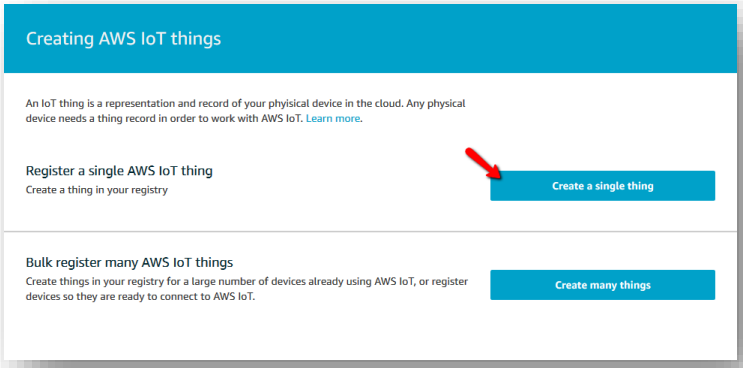
14. Create a new Thing

    14.1. Click the "**Create**" button to start process

    If you do not have any IoT things registered in your account, the **You don't have any things yet** page is displayed. If you see this page, choose **Register a thing**. Otherwise, choose to **Create**.

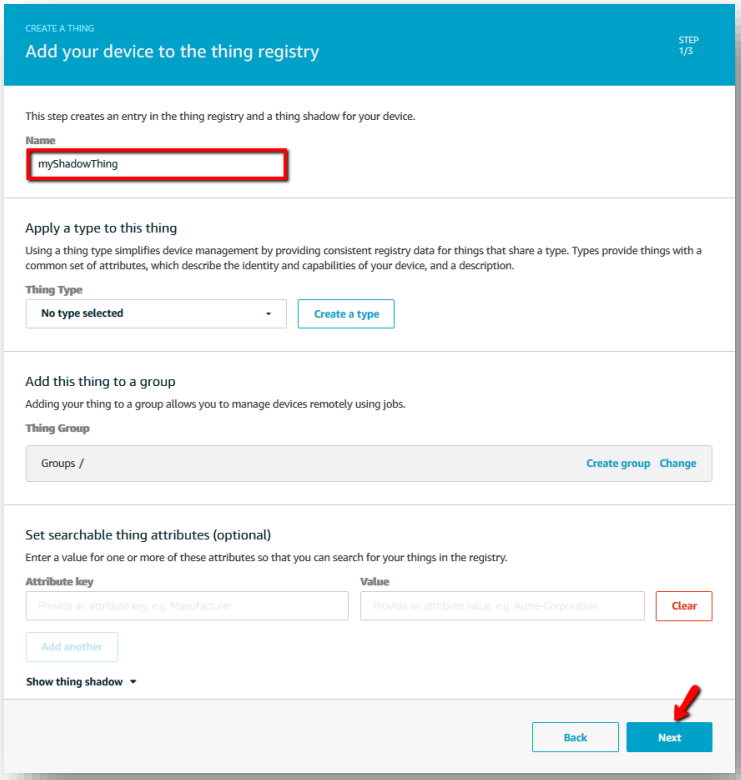


14.2. Click the "**Create a single thing**" option



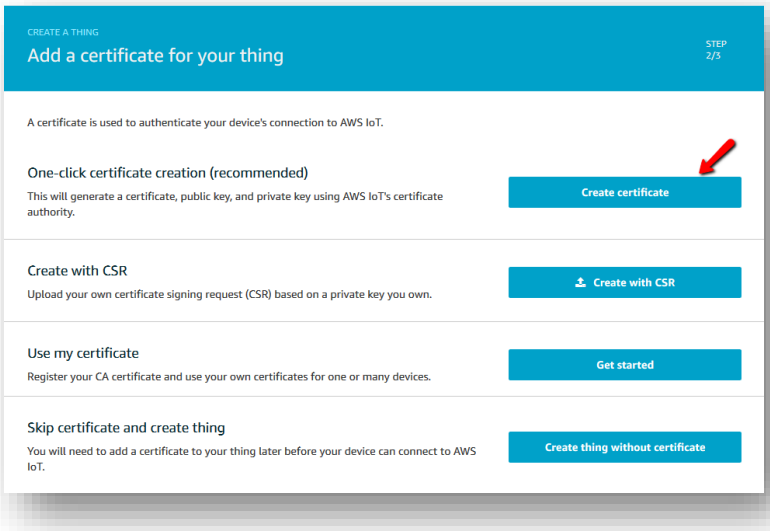
14.3. Provide a unique name for your thing (myShadowThing1, myShadowThing2 ,… myShadowThingX). You will use this name throughout the class.

14.4. Scroll-down and select **Next**



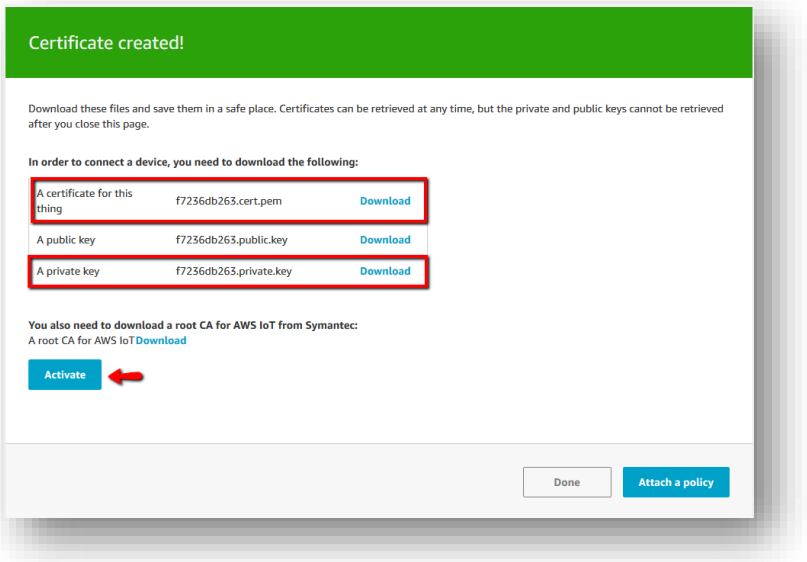
15. Generate security certificates for your Thing

15.1. Use the “**One-click certificate creation**” option to by clicking the “**Create certificate**” button.



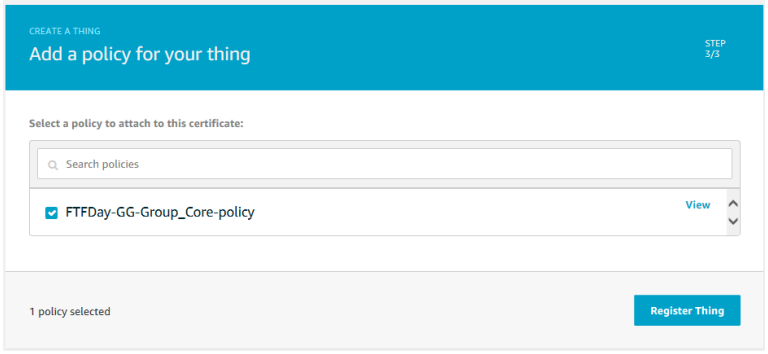
15.2. **Important #1**: Download the certificate (*.pem) and private key (*.private.key)

15.3. **Important #2**: Click the **Activate** button



15.4. Next proceed to the "**Attach a policy**" step, selecting the "**FTFDay-GG-Group_Core-policy**"

16. Finish the process by selecting "**Register Thing**"

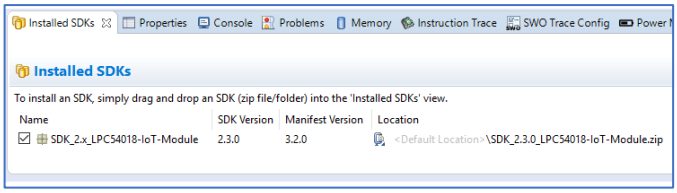


Updating the Client Credentials

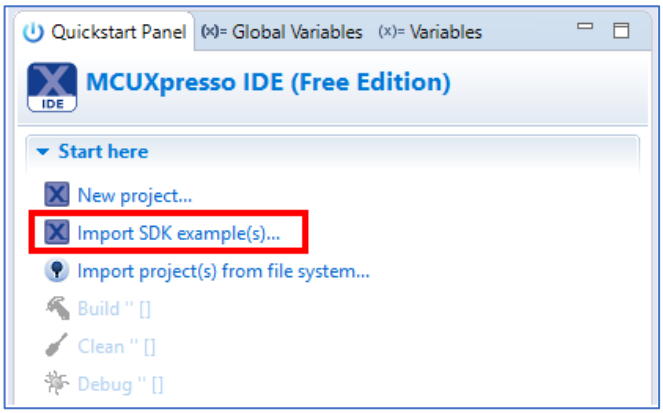
17. Import the **aws_examples/aws_led_wifi_qspi_xip** SDK examples into the MCUXpresso IDE

    17.1. Open MCUXPresso IDE v10.2 (icon on desktop)

    17.2. Ensure that the **SDK_2.x_LPC54018** is installed and is **unzipped**



    17.3. Use the Quickstart Panel to **Import SDK example(s)…**

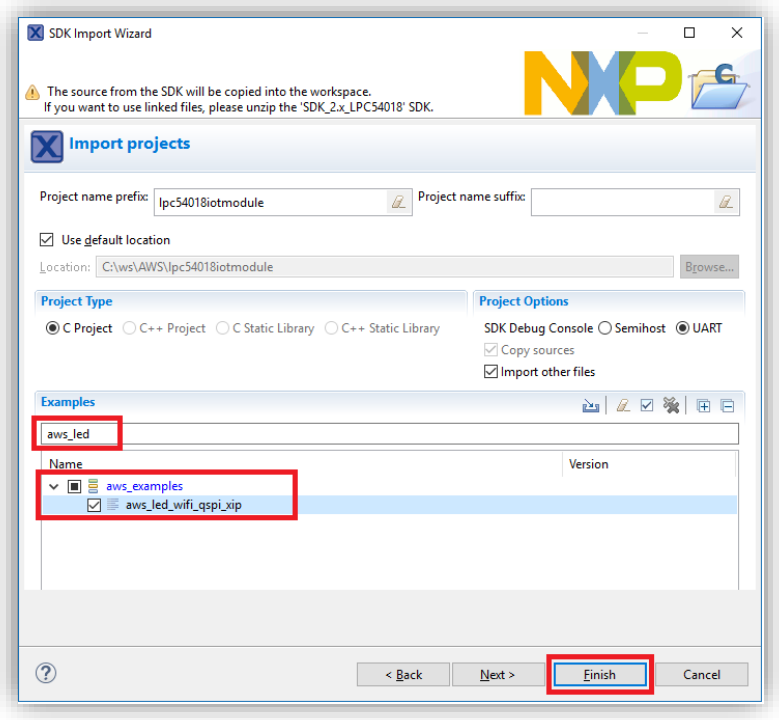


    17.3.1. Search for “**iot**” in the available board search box

17.3.2. Click on the image of the **lpc54018iotmodule**, and select **Next**

17.3.3. Expand the **aws_examples** category and place a checkmark next to **asw_led_wifi_qspi_xip**

17.3.4. Select **Finish** to import the SDK example project

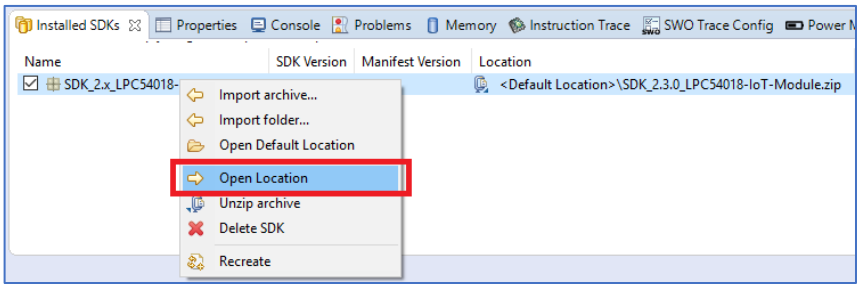


18. Explore the structure of the project and view the **readme.txt** file in the doc subfolder

18.1. Middleware subfolders: amazon-freertos, mbedTLS, usb, wifi_qca

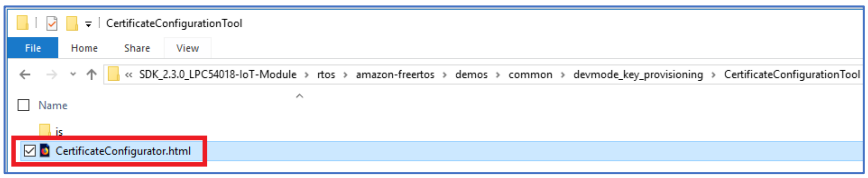
19. Generate the client credential keys source file

19.1. Right-click on the **SDK_2.x_LPC54018** row in the "Installed SDKs" panel and select **Open Location**



19.2. In the resulting window browse to:
..\rtos\amazon-freertos\demos\common\devmode_key_provisioning\CertificateConfigurationTool

19.3. Double-click the **CertificateConfigurator.html** to open in a web browser

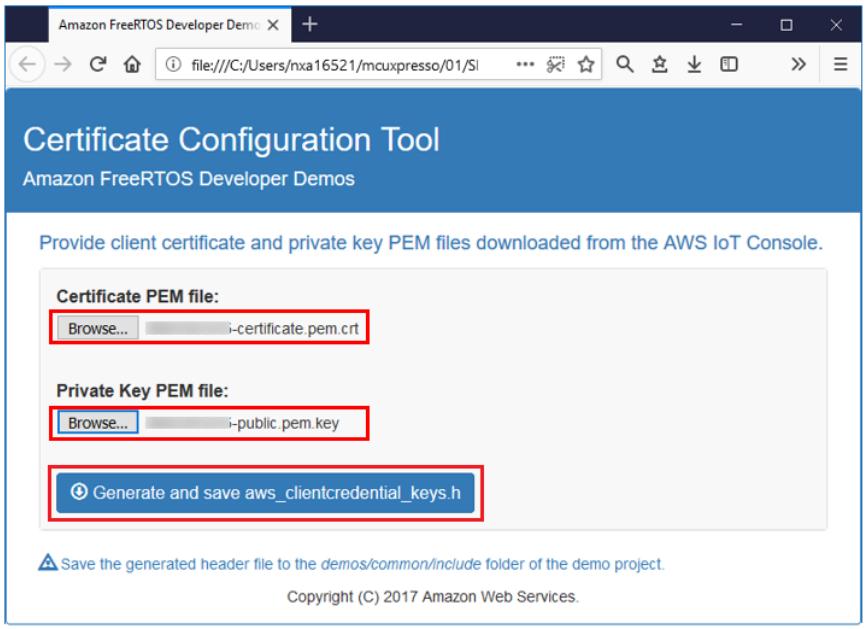


19.4. Generate the **aws_clientcredential_keys.h** file

19.4.1. Browse to and select the **Certificate PEM file (*-certificate.pem)** that you downloaded in Lab 1

19.4.2. Browse to and select the **Private Key PEM file (*-private.pem.key)** that you downloaded in Lab 1

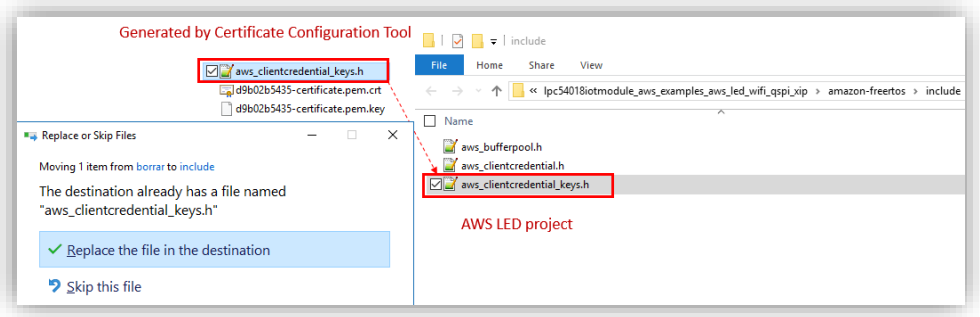
19.4.3. Click the "**Generate and save**" button, which is also be save to the "Downloads" library



19.5. Open a file explorer window and locate the **aws_clientcredential_keys.h** file in the Downloads library

19.6. Use **CTRL-C** to copy the file **aws_clientcredential_keys.h**

19.7. Return to the MCUXpresso IDE and select the **amazon-freertos -> include** subfolder. With the **include** folder selected press **CTRL-V** to paste the file into directory. You should get an Overwrite confirmation message, select **Overwrite**



19.8. Double-click the **aws_clientcredential_keys.h** in the IDE to confirm that is contains a large block of random characters in the certificate sections

```
 9 static const char clientcredentialCLIENT_CERTIFICATE_PEM[] =
10 "-----BEGIN CERTIFICATE-----\n"
11 "MIIDWTCCAkGgAwIBAgIUbGczOYf7weX4DdHSh5vsagopXBowDQYJKoZIhvcNAQEL\n"
12 "BQAwTTFLMEkGA1UECwxCQW1hem9uIFdlYiBTZXJ2aWNlcyBPPUFtYXpvbi5jb20g\n"
```

20. Obtain the AWS endpoint address

20.1. Return to the AWS IoT tab of your browser

20.2. Click on the name of the Thing you just created
Note: every class participants Thing will appear on this page, ensure you are only selecting your's throughout this lab.

20.3. Select the **Interact** section and copy the **REST API Endpoint** from the resulting screen: **a3**********.iot.us-east-1.amazonaws.com**

20.4. Highlight and copy this string using CTRL-C

21. Configure additional Client Credentials

21.1. Return to the MCUXpresso IDE application

21.2. Double-click the **aws_clientcredential.h** file in the IDE Project Explorer (**amazon-freertos/include** subfolder) to open it in the IDE editor

21.3. Paste the REST API Endpoint (also referred to as the **MQTT_Broker_Endpoint**) into **line 38**, keeping the quotation marks

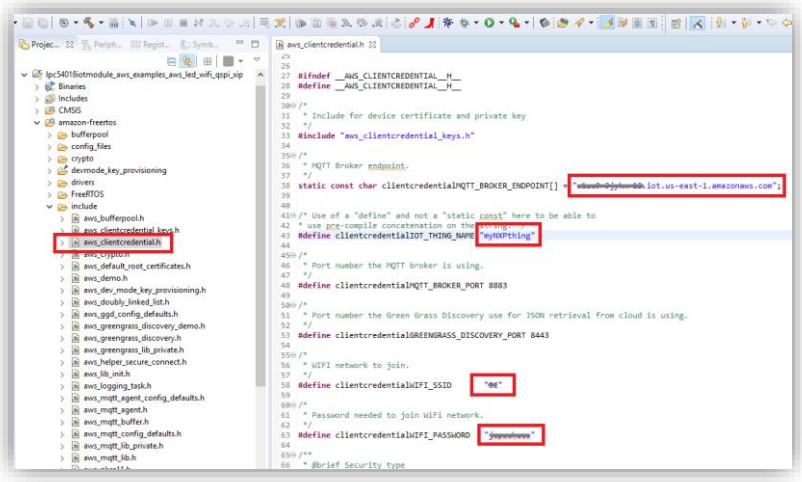
21.4. Update the **IOT_THING_NAME** on Line 43 to match the name of your Thing exactly, keeping the quotation marks

21.5. Update WiFi SSID and Password

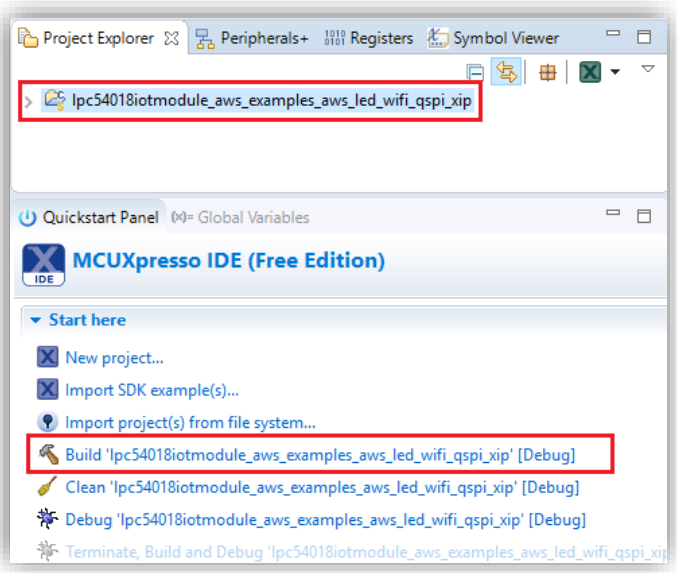
21.5.1. Line 58: replace "Paste WiFi SSID here." with **“nxp-aws-wifi”**

21.5.2. Line 63: replace "Paste WiFi password here." with “**nxp-aws-pw**”

21.6. Save changes to the **aws_clientcredential.h** file.



22. From the **Quickstart Panel** select **Build** and confirm there are no errors once the build completes.




```
Building target: lpc54018iotmodule_aws_led_wifi_qspi_xip.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -L"C:\Users\nxa08675\Documents\MCUXpressoI
Memory region         Used Size  Region Size  %age Used
     BOARD_FLASH:      361548 B        16 MB      2.15%
           SRAMX:       57332 B       192 KB     29.16%
    SRAM_0_1_2_3:       96352 B       160 KB     58.81%
        USB_RAM:        5584 B         8 KB     68.16%
Finished building target: lpc54018iotmodule_aws_led_wifi_qspi_xip.axf

make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size "lpc54018iotmodule_aws_led_wifi_qspi_xip.axf"; # ar
   text    data     bss     dec     hex filename
 361548       0  102960  464508   7167c lpc54018iotmodule_aws_led_wifi


23:06:52 Build Finished (took 2m:5s.345ms)
```

General Debugging Procedures: Important

Due to the nature of the IoT module, this application uses a USB CDC VCOM to send output to the screen. While debugging the USB task are also subject to the runtime controls. Additionally, the USB needs to enumerate with the host PC each time the debug session is started. This can lead to a sense that the application is not responsive.

Follow the steps below regarding Tera Term and the IDE Debug Session to ensure the application runs as expected.

1. Two USB cables should be connected to the development board.
   1.1. One connected to the **Debug Link** port of the Baseboard
   1.2. One connected to the microUSB connector on the **module** itself
2. Use the **Quickstart Panel** to **Build** the application, ensure there are no compile errors
3. Use the **Quickstart Panel** to **Debug** the application
   3.1. The initial debug session will launch the Probe Discovery window and will recognize the **J-Link LPCXpresso V2 debug probe**
4. Use the **Resume** (play) button to start the applciation
5. For first debug session: open Tera Term (or switch to it for subsequent debug sessions)
   5.1. Create a new connection, selecting the **Serial port: COM xx: USB Serial Device**
   5.2. Tera Term will likely NOT appear responsive initially, but will eventually connect and display output captured from the beginning.
6. For **subsequent** debug sessions: switch to Tera Term which should already be open and configure to correct COM setting
   6.1. Tera Term **may** be unresponsive and target application may appear unresponsive as well. **If so:**
      6.1.1. **Wait** for the Tera Term window to clear the "Not Responding" status, screen will still not be updating
      6.1.2. Select "Edit -> Clear Screen" if desired
      6.1.3. Select "**File -> Disconnect**" (alt-i)
      6.1.4. Select "**File -> New Connection**" and "OK" (alt-n)
      6.1.5. Terminal will update with the buffer of console output and the application will continue running as expected.
7. The IDE "Restart" button that be used to restart the application without reflashing.
   7.1. Press the "Restart" icon 
   7.2. Jump to Step 4 above

Lab 2 – Debugging the AWS LED Application

Wi-Fi Connection

1. Place a break point on **Line 320** and **Line 334** of **led_control.c**
2. Run the Application
   2.1. Use the **Quickstart Panel** to **Build** and **Debug** the application
   2.2. Start the application using the **Resume** (play) button. Follow the guidelines on the previous page to ensure you have a working console and that the debugger progresses to the breakpoint set above
3. Review the Console output to ensure you have connected to the WiFi Access point and have acquired an IP address 192.168.12.xxx

Shadow Client Connection to AWS

4. Use the Run-controls to Step-Into  Line 320 "ShadowClientInit"
   4.1. Step thru the "Shadow Client Init" function noting the output on the console
      4.1.1. Line 159: **SHADOW_ClientCreate**
      4.1.2. Line 176: **SHADOW_ClientConnect**
         - Connect to AWS using the REST API Endpoint and resolves the endpoint into an IP address
         - Establishes a client connection to AWS
      4.1.3. Line 184: **SHADOW_RegisterCallbacks**
         - Registers the delta callback function and subscribes to the callback topic
5. Press the **resume** button to jump to the next breakpoint (Line 334)
6. Open Shadow Page
   6.1. Click on your "**Thing**" on the IoT Device Management page
   6.2. Click on "**Shadow**" to view the cloud copy of the shadow

7. Return to the IDE, step thru the “ShadowMainTask” function noting the output on the console

   7.1. Line 334: **SHADOW_Delete**

   - Maintains subscription to the default shadow topics: Accepted, Rejected
   - Publishes the Delete command to the operations topic
   - Note the Shadow Document on the web page is now deleted

   7.2. Line 344: **SHADOW_Update**

   - Cleans up subscriptions: Accepted, Rejected
   - Publishes the Update: creating the default Shadow document based on the “prvGenerateShadowJSON” function

8. Press the **resume** button to continue the application

Response to Delta Messages

9. Clear existing break points on **Lines 320** and **334** of **led_contol.c**
10. Add breakpoint on **Line 96** and **Line 360** of **led_control.c**
11. Switch back to web browser and go to the **Shadow page** of your Thing
12. Manually edit the Shadow Document

    12.1. Select the **Edit** action on the page

    12.2. Carefully update the desired LEDstate to 1

    12.3. Press **Save**, IDE should have hit breakpoint at line 96

    12.3.1. This function receives the delta message and points the contents of the message to a message queue

13. Press play to jump to the next breakpoint
14. Step thru code

    14.1. Line 362: **processShadowDeltaJSON** – uses JSON parsers to extract the desired state of the led, setting the **parsedLedState** global variable to the desired value.

    14.2. Line 370 or 376: LED on module will change

    14.3. Line 380: **prvReportShadowJSON** – generates the update message with the new reported value

    14.4. Line 381: **SHADOW_Update** – publishes the update message

    14.5. Line 384: **SHADOW_ReturnMQTTBuffer** – returns the dedicated MQTT buffer

15. Clear breakpoints on **Lines 96** and **360** of **led_contol.c**
16. Press the resume button to resume normal operation with not set breakpoints
17. Continue changing the Shadow Document on the web and see the edit reflects on the LED of the module.

Wifi Router Info:

Router name : nxp_iot_aws

Password : nxp_iot_demo